

Audio-Chat

Diogo Brandão Ferreira[a48530]

José Nuno Marinho Carvalho[a48542]

Instituto Politécnico de Bragança

Este miniprojecto consiste em criar um sistema de áudio-chat entre dois clientes, utilizando o “dashboard” do Node-RED. Cada cliente poderá enviar mensagens de áudio de até 10 segundos para o outro cliente, e as mensagens devem ser criptografadas antes de serem enviadas. Além disso, as mensagens enviadas pelos clientes devem ser convertidas em texto e enviadas por meio do aplicativo “Telegram”, onde cada cliente terá seu próprio registo das mensagens.

Para implementar o sistema, foi necessário instalar a biblioteca “node-red-ui-microphone” e garantir as permissões de acesso no navegador para gravar o áudio do microfone. Será possível ter acesso aos 10 últimos áudios enviados, descartando o primeiro áudio quando um novo for enviado. E também a biblioteca “node-red-contrib-telegrambot” para realizar a comunicação entre o “Node-RED” e o “Telegram”. Os recursos necessários para a implementação do projeto incluem os conceitos e ferramentas apresentadas nas aulas práticas e teóricas de Internet das Coisas (IoT), como o protocolo MQTT e o broker, o Node-RED e mecanismos de segurança.

Keywords: Telegram · Node-RED · MQTT.

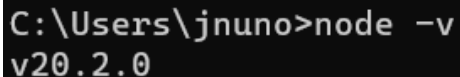
1 Node-RED - Configuração

1.1 Node.JS

Para começarmos a trabalhar no node-red é necessário ter o Node.js instalado que é necessário para executar o Node-RED. Para verificar se o Node.js está instalado executamos o seguinte comando na linha de comandos:



```
node -v
```



```
C:\Users\jnuno>node -v  
v20.2.0
```

Imagem 1 e 2 . Verificação da versão do Node

De seguida após confirmação da existência do Node.js, prosseguimos para fazer também na linha de comandos o uso do seguinte comando:



```
npm install -g --unsafe-perm node-red
```

Imagem 3: Comando para instalação do node-red

Agora já reunimos todas as condições para aceder ao localhost: <http://127.0.0.1:1880/> e começar a implementar o nosso projeto.

2 Instalação das bibliotecas necessárias do node-RED(node-red-contrib-telegrambot e node-red-ui-microphone)

Para fazermos a instalação das bibliotecas sugeridas, pesquisamos pela opção "Manage palette" no canto superior direito do editor do Node-RED, depois na guia "Install" pesquisamos pelas bibliotecas pretendidas, e procedemos á instalação das mesmas

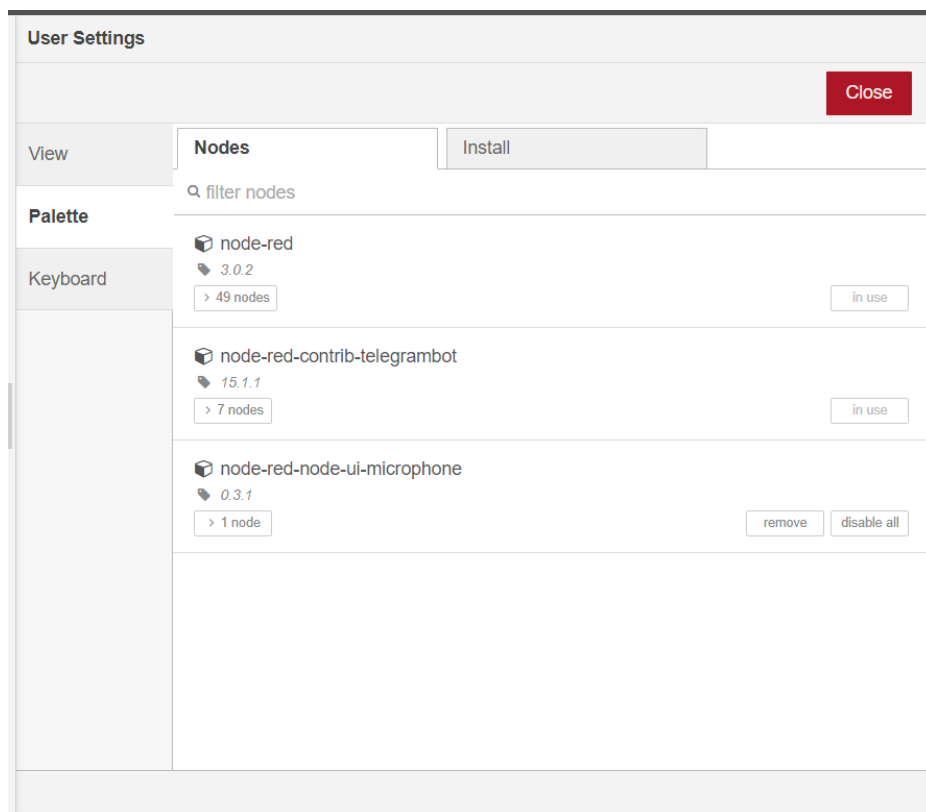


Imagem 4: Imagem do Node-RED para instalação das bibliotecas

Passando a uma breve explicação de qual a função de cada node, começamos pelo node "node-red-contrib-telegrambot". Com o node do telegrambot devidamente configurado, como iremos também detalhar abaixo, permite-nos interagir com o Telegram por meio do Node-RED, sendo possível enviar mensagens, receber atualizações do Telegram e realizar outras ações com um Bot do Telegram que foi já criado por nós.

Em relação ao node do microfone(node-red-ui-microphone) é possível gravar e reproduzir áudio dentro do painel do Node-RED. O widget do microfone permitirá que seja inicializada a gravação de áudio e a reproduza conforme necessário.

Agora, de uma forma mais detalhada, passaremos a descrever qual a função de cada node de uma forma mais concreta no nosso projeto.

2.1 Node-red-contrib-telegrambot

-Telegram Receiver Node: Permite receber mensagens e atualizações do Telegram. Sendo possível configurar o node para receber mensagens de texto, imagens, documentos, localização e outros tipos de conteúdo. No nosso caso iremos configurar para receber áudios.

-Telegram Sender Node: Com este node, é possível enviar mensagens, fotos, documentos, áudio, localização e outros tipos de conteúdo para usuários ou grupos do Telegram. Podemos definir dinamicamente os destinatários e o conteúdo da mensagem usando informações do fluxo



Imagem 5: Nós da biblioteca do telegram

2.2 Node-red-node-ui-microphone

O nó fornece um único botão que, ao ser clicado, começará a capturar o áudio.

Este nó fornece um widget no painel que, quando pressionado, começará a capturar o áudio ou o reconhecimento de fala. O botão pode ser configurado em dois modos para o modo de captura de áudio, que no nosso caso vai ser de reconhecimento de voz.

O clique começa a capturar o áudio e continua a capturar o áudio até que o botão seja pressionado novamente ou atinja a duração máxima configurada que no nosso caso é de 10 segundos. O botão pode ser configurado também para gravar apenas enquanto o botão estiver pressionado. Para o modo de captura de áudio, o áudio é capturado no formato WAV e publicado pelo nó como um objeto Buffer. Isso pode ser gravado diretamente em um arquivo ou passado para qualquer outro nó que espera dados de áudio

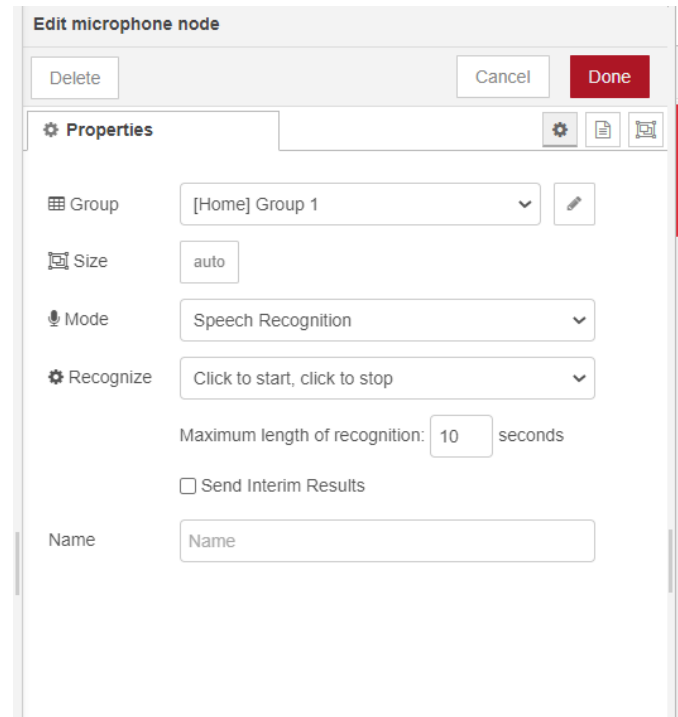


Imagem 5: Configuração do nó do microfone

3 Dashboard do Node-RED.

Para conseguirmos ter um áudio-chat eficiente com a correta aplicação do uso do microfone é necessário um painel de controlo (dashboard) adicionado ao nosso fluxo do Node-RED. Primeiramente é necessário a instalação do **node-red-dashboard**, sendo possível configurar se necessário o node “ui_tab” e o node “ui_group” de forma a agrupar os elementos desejados para o painel. Depois é possível também adicionar widgets para exibir informações relevantes tais como “como ui_text, ui_chart, ui_gauge, ui_table”. Por fim fazemos o “deploy” do nosso “flow” para implementar a dashboard.

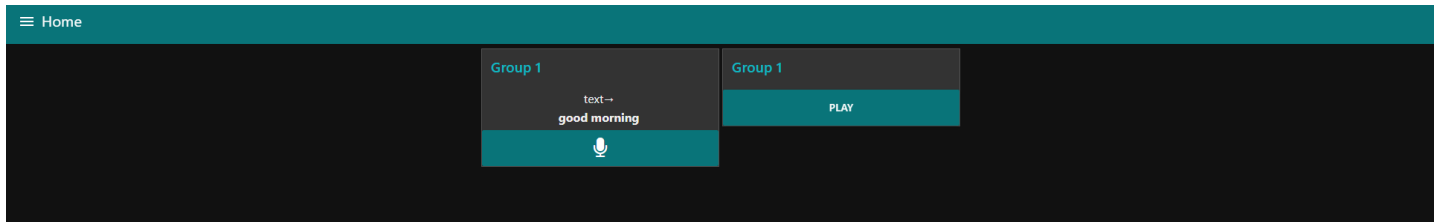


Imagem 6: Imagem do nosso dashboard agrupados no “home”

No dashboard implementado é feito o registo do input do microfone, onde logo após receber o input é encriptada por um cliente, descriptada e finalmente enviada para o telegram. No dashboard temos dois botões (um para o registo do input do microfone, e outro para ser reproduzido pelas colunas do computador o áudio desse input). É possível também observar a descriptação do áudio do microfone para “plain-text”.

4. Clientes MQTT

É necessário também a implementação e configuração dos Clientes MQTT A e B.

Os clientes A e B são referências aos dois participantes do áudio-chat no Node-RED.

Estes representam as entidades que se comunicam entre si por meio do fluxo do Node-RED.

É para isso necessário adicionar dois nós MQTT ao fluxo do Node-RED, um para o cliente A e outro para o cliente B, configurando cada nó MQTT com as informações de conexão ao broker MQTT, como endereço do servidor, porta e tópicos relevantes. Utilizamos nodes adequados para gravação e reprodução de áudio no Node-RED, conectando esses nós aos nós MQTT correspondentes dos clientes A e B. Isso permitirá que os clientes gravem e enviem áudios para o tópico MQTT associado ao outro cliente, e também recebam e reproduzam os áudios recebidos pelo tópico MQTT.

Para garantir a segurança da comunicação, adicionamos nós de criptografia adequados para criptografar e descriptar os áudios trocados entre os clientes A e B. Isso incluiu a geração e o compartilhamento de chaves de criptografia, que usam algoritmos criptográficos seguros.

Utilizando o pacote node-red-contrib-telegrambot , configuramos os nós de comunicação entre o Node-RED e o Telegram para cada cliente. Configuramos os nós para enviar e receber mensagens de texto convertidas a partir dos áudios gravados pelos clientes A e B.

Por fim para exibir os últimos 10 áudios trocados entre os clientes A e B no painel do Node-RED, é necessário armazenar os áudios numa estrutura de dados (como um array) e atualizar sempre que um novo áudio for recebido.

The screenshot shows the 'Edit mqtt out node' configuration window. It has a title bar with 'Delete', 'Cancel', and 'Done' buttons. Below the title bar is a 'Properties' tab with a settings icon, a document icon, and a refresh icon. The configuration fields are: 'Server' (broker.mqtt-dashboard.com:1883), 'Topic' (audio-chat), 'QoS' (dropdown), 'Retain' (checkbox), and 'Name' (Cliente A). A tip box at the bottom states: 'Tip: Leave topic, qos or retain blank if you want to set them via msg properties.' At the very bottom, there is an 'Enabled' checkbox.

The screenshot shows the 'Edit mqtt in node' configuration window. It has a title bar with 'Delete', 'Cancel', and 'Done' buttons. Below the title bar is a 'Properties' tab with a settings icon, a document icon, and a refresh icon. The configuration fields are: 'Server' (broker.mqtt-dashboard.com:1883), 'Action' (Subscribe to single topic), 'Topic' (audio-chat), 'QoS' (2), 'Output' (auto-detect (parsed JSON object, string or buffer)), and 'Name' (Cliente B).

Imagem 6 e 7 : Configuração dos Clientes A e dos Clientes B , usando os nodes “mqtt in e mqtt out”

Ou seja, adicionamos um nó MQTT na função de gravação de áudio do cliente, configuramos o broker com aquele que temos vindo a utilizar nas aulas práticas, e definimos o tópico como áudio-chat, sendo obviamente algo em comum com os dois clientes.

Desta forma o cliente A consegue estabelecer uma ligação com o cliente B e vice-versa.

5. Nodes de Encriptação e de Desencriptação.

Para adicionar criptografia AES (Advanced Encryption Standard) aos áudios trocados entre os clientes A e B no Node-RED, escolhemos utilizar os nodes de criptografia AES disponíveis no Node-RED. A criptografia AES é um algoritmo amplamente utilizado para proteger a segurança de dados. Instalamos o pacote “node-red-contrib-simple-crypto” no Node-RED. Este pacote fornece nodes de criptografia simples, incluindo encriptação e desencriptação que vão servir para se relacionarem com os áudios.

Adicionamos o node "Encrypt" (Encriptação) ao fluxo do Node-RED, conectando-o ao nó que recebe os áudios dos clientes A e B.

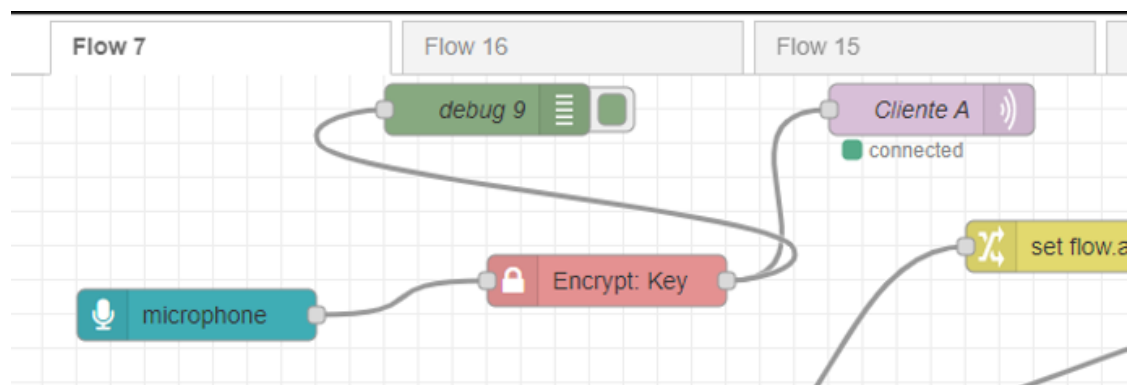


Imagem 8: Integração do Nó Encrypt no nosso flow.

Configuramos, portanto, o node "Encrypt" com a chave de criptografia adequada. A chave de criptografia é uma sequência de caracteres que precisamos de gerar e compartilhar com antecedência entre os clientes A e B. Certifique-se de manter essa chave em segurança.

Conectamos o node "Encrypt" ao nó MQTT correspondente para enviar o áudio encriptado ao cliente recetor.

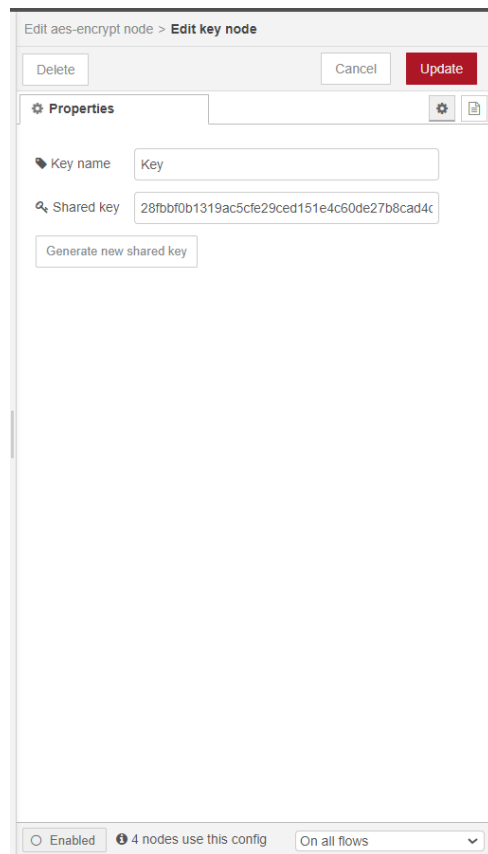


Imagem 9: Configuração do nó "Encrypt" com a inclusão de uma chave gerada.

No cliente recetor, adicionamos o node "Decrypt" (Desencriptação) ao fluxo do Node-RED, conectando-o ao nó MQTT que recebe o áudio encriptado.

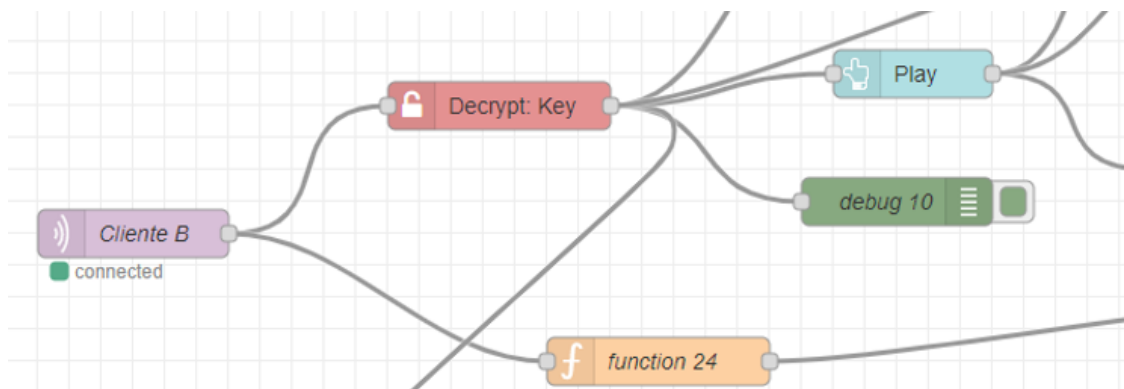


Imagem 10: Integração do nó Decrypt no nosso flow.

Configuramos depois o node "Decrypt" com a mesma chave de criptografia utilizada para a encriptação e conectamos o node "Decrypt" ao node de reprodução de áudio para reproduzir o áudio descriptado para depois enviá-lo para o “telegram”.

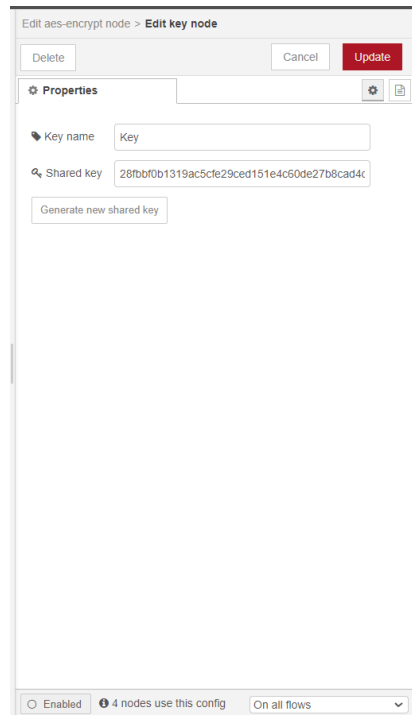


Imagem 11: Configuração do nó decrypt

Desta forma, os áudios serão encriptados antes de serem enviados pelo cliente A e descriptados pelo cliente B antes de serem reproduzidos. A utilização da mesma chave de criptografia nos dois clientes garante que somente eles serão capazes de descriptar os áudios corretamente.

6. Configuração do bot do Telegram

Prosseguindo para uma das etapas mais importantes da elaboração do miniprojecto, temos agora a configuração do nosso Bot do telegram.

Primeiramente necessitamos da instalação do nó: “node-red-contrib-telegrambot” no nosso Node-RED.

No editor do Node-RED, é necessário adicionar um nó "Telegram Receiver" para receber mensagens do Bot do Telegram.

Iremos, contudo, necessitar de criar um novo Bot ou usar um Bot existente na plataforma do Telegram. Para a criação do Bot usamos os seguintes comandos como pode ser visível nas imagens abaixo

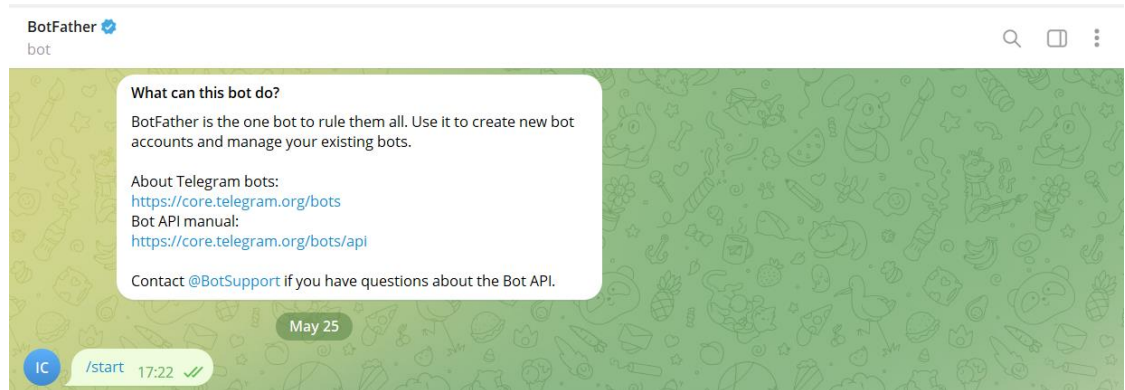


Imagem 12: Solicitar o “/start” ao BotFather no Telegram.

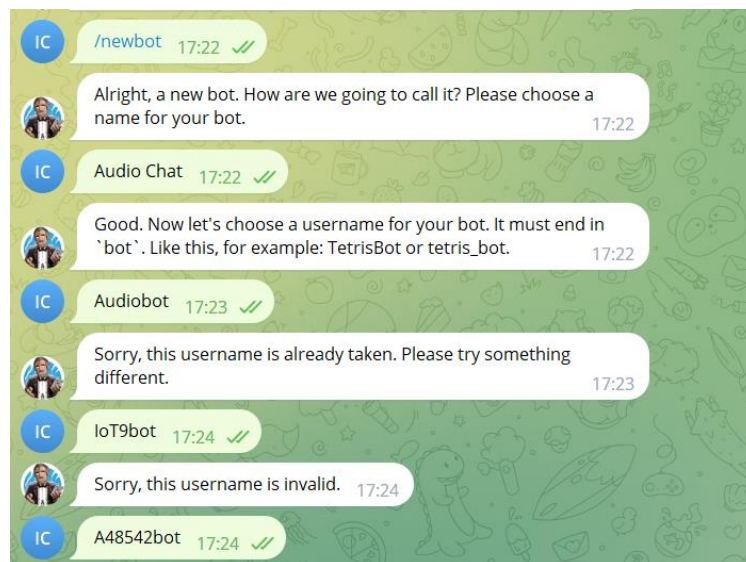


Imagem 13: Criação do novo Bot com o nome “Audio Chat” e com o username “A48542bot”

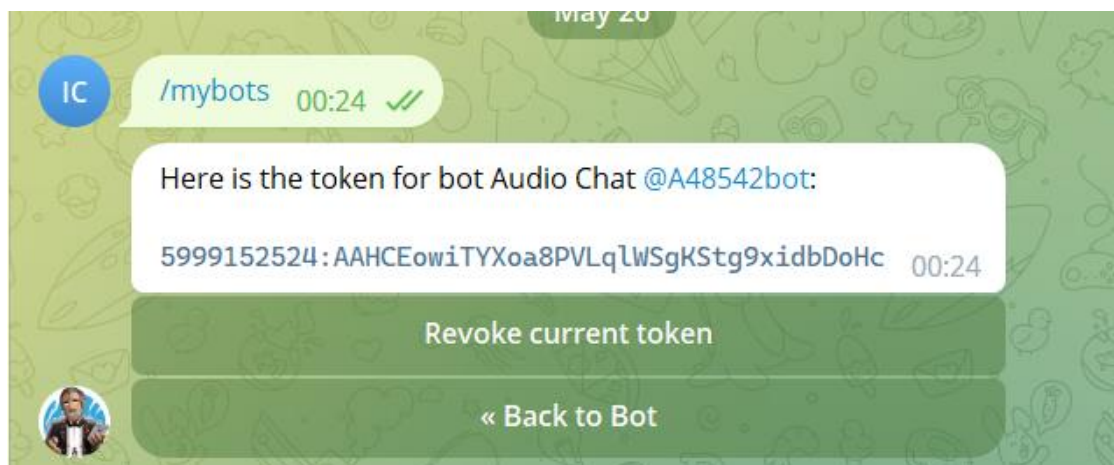


Imagem 14: Token do nosso Bot (A48542)

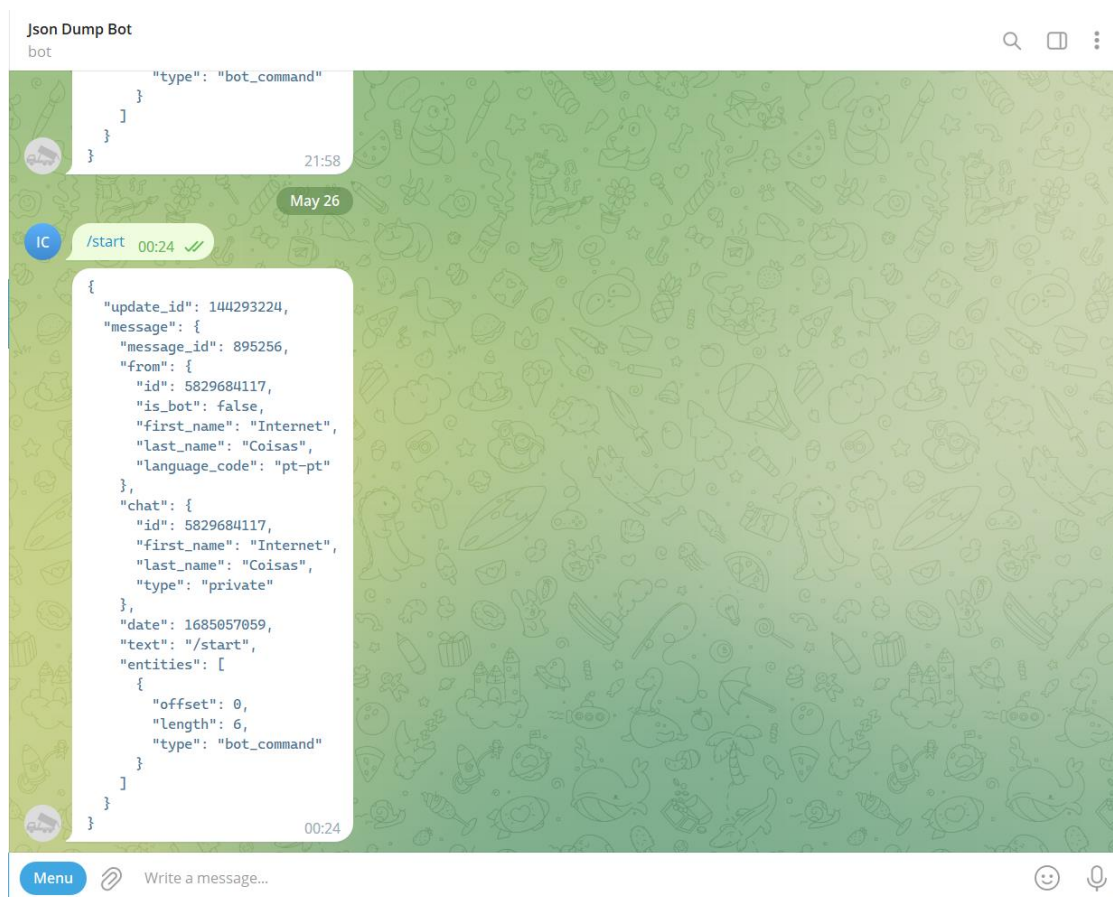


Imagem 15: “Json Dump” do nosso Bot com informações relativas a chatId, primeiro nome, ultimo nome e tipo.

Ou seja abrimos o Telegram no nosso computador, pesquisando pelo BotFather, criamos um novo Bot com o comando: “/newbot” , escolhemos o nome e o username para o nosso Bot , sendo depois fornecida um Token da API para o Bot.

No Node-RED, já com o Bot criado, no nó "Telegram Receiver" e no nó “Telegram Sender” inserimos as informações do nosso Bot.

The image shows the 'Edit telegram bot node' configuration panel in Node-RED. The panel has a title bar 'Edit sender node > Edit telegram bot node' and buttons for 'Delete' and 'Cancel'. Below is a 'Properties' section with the following fields:

- Bot-Name:** A48542bot
- Token:** 5999152524:AAHCEowiTYXoa8PVLqIWSgKStg9xidbDoHc
- Tip:** If you don't have a token yet, you can create a new one here: [@BotFather](#).
- Users:** (Optional list of authorized user names e.g.: hugo,sepp,egon)
- ChatIds:** 5829684117
- Server URL:** (Optional URL for proxying and testing e.g.: https://api.telegram.org)
- Update Mode:** Polling (dropdown menu)
- Polling Options:**
 - Poll Interval:** 300

Imagem 16: Configuração do nó “Telegram sender node” com as configurações do Bot

Depois foi só conectar o nó "Telegram Receiver" e o nó “Telegram Sender” aos outros nós do fluxo seguindo a nossa lógica.

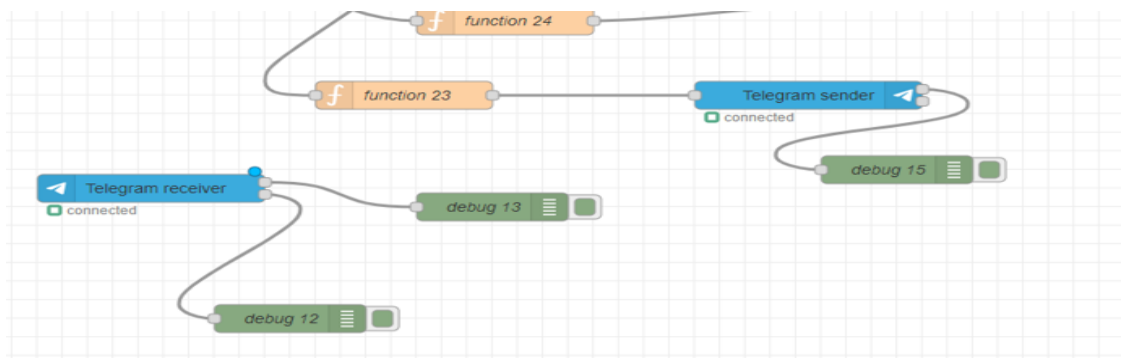


Imagem 17: Nós do telegram no nosso fluxo

7. Fluxo do Node-RED explicado

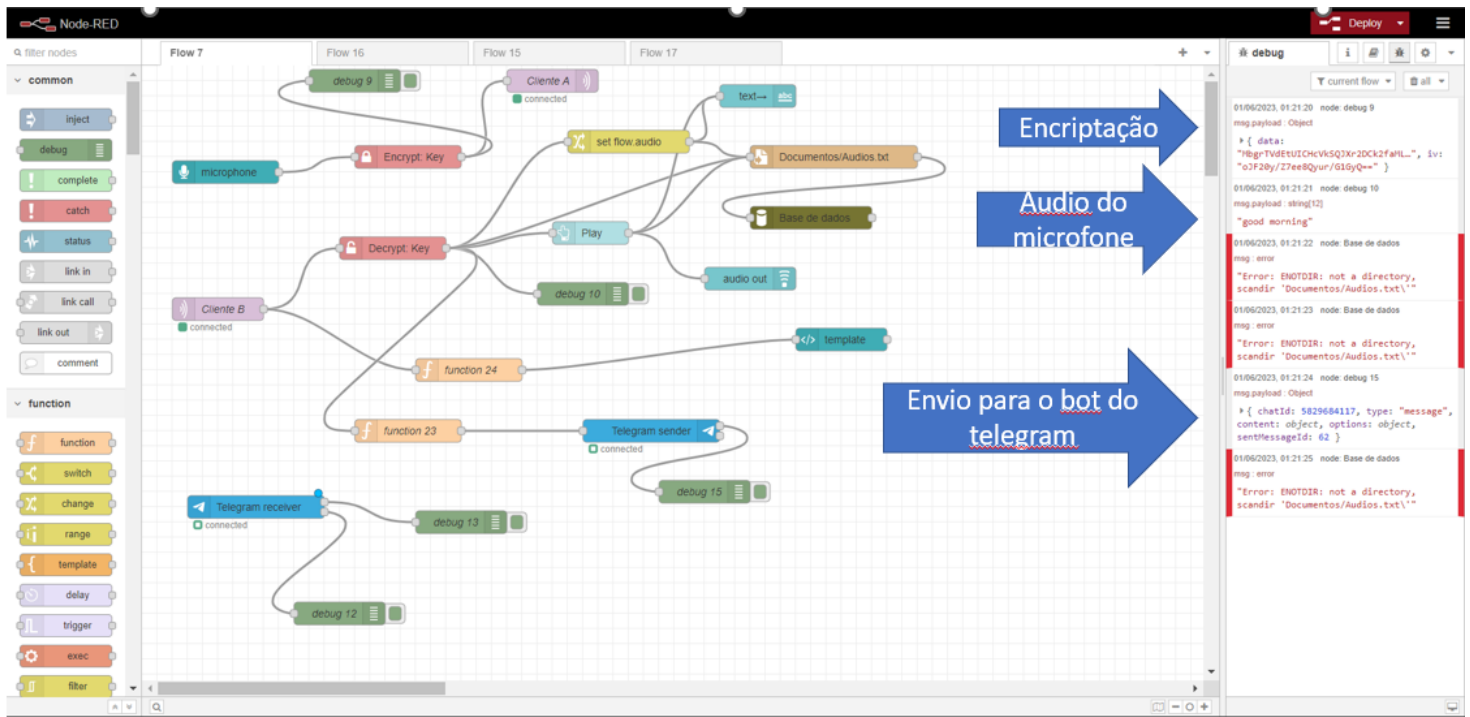


Imagem 18: O Fluxo do Node-RED implementado

Explicando agora o nosso fluxo do Node-RED de uma forma mais geral, detalhada e por passos:

1. Introdução do microfone em modo de reconhecimento de fala estando conectado a um nó “AES-Encrypt” que irá fazer a encriptação do output da voz, que de seguida se irá conectar ao cliente A.

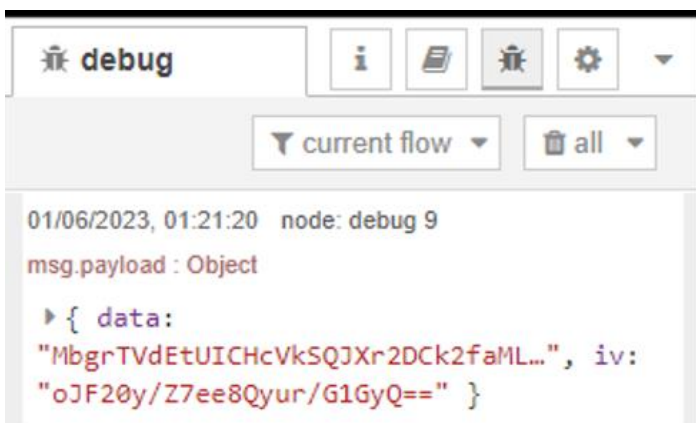


Imagem 19: Output do Microfone encriptado usando AES

2. Cliente B recebe os dados do Cliente A(Output do Microfone encriptado) que passa pelo nó “Decrypt” de forma a obtermos o output do microfone em texto. Este por sua vez está conectado a um nó denominado “play” que por sua vez está conectado a um nó “áudio out” que permite a reprodução em formato áudio do output do microfone sempre que pressionamos o botão “PLAY” na nossa dashboard.

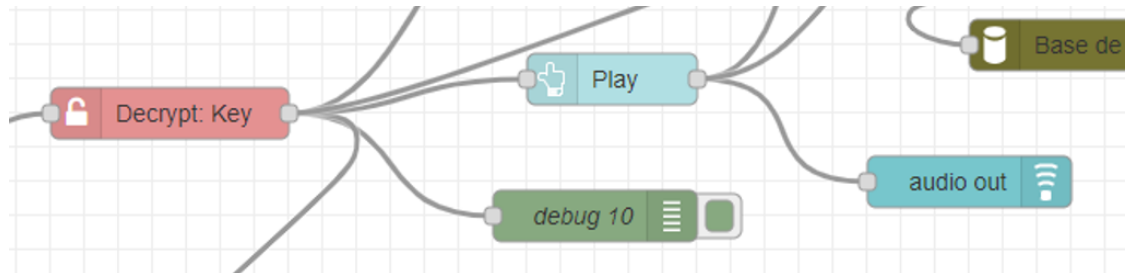


Imagem 20: Output descriptado conectado ao nó: “Play” que de seguida está conectado ao nó “áudio out”

```
01/06/2023, 01:21:21 node: debug 10
msg.payload : string[12]
"good morning"
```

Imagem 21: Output do microfone descriptado pelo nó “AES Decrypt”

Importante referir que o nosso nó “Decrypt” está também conectado a um nó denominado “Set Flow Audio” que serve para estabelecer uma relação do áudio com o “msg.payload” que consequentemente está conectado a um nó “text” que nos permite mostrar em texto o output do microfone na dashboard.

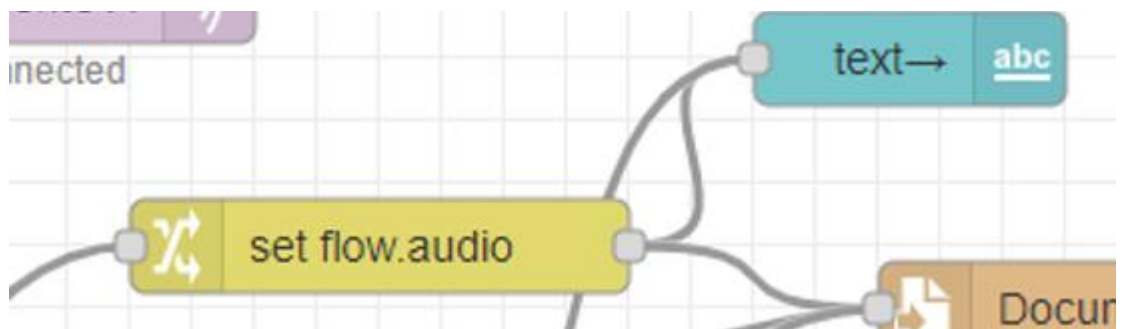


Imagem 21: Nós “set flow.audio” e “text”

3. O Nó “Cliente B” está também conectado a um nó “function24” que por sua vez está conectado a um nó “template” que tem como objetivo armazenar os últimos 10 áudios e mostrar os mesmos na nossa dashboard.

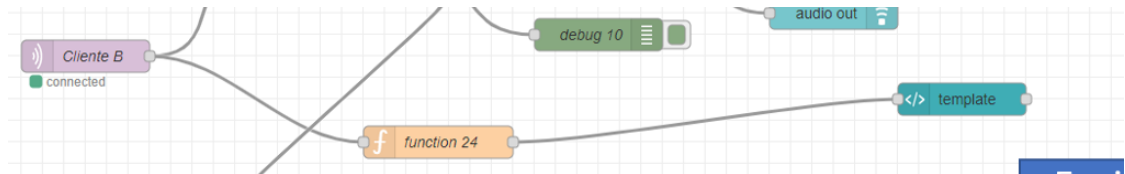


Imagem 22: Nós “Cliente B”, “function24” e “template”

```
1  const maxAudios = 10;
2  const audio = msg.payload; // Substitua pela propriedade corr
3
4  let audios = context.get("audios") || []; // Obtém a lista de
5  audios.unshift(audio); // Adiciona o novo áudio no início da
6  audios = audios.slice(0, maxAudios); // Limita a lista aos úl
7
8  context.set("audios", audios); // Atualiza o contexto com a l
9
10 return msg;
11
```

Imagem 23: Função 24 que está conectada ao cliente B

Ou seja, esta função 24 define como variável “maxAudios” = 10 , significando que tem como máximo de áudios possíveis a serem armazenados como 10. Também definida como variável temos áudio que vai estar ligada ao output que o “msg.payload” apresenta. De seguida obtêm-se a lista de áudios, e sempre que um novo áudio “chega” este é adicionado ao início da lista.

4. Também conectado ao nó “Decrypt” temos um nó “função 23” que está respectivamente conectado ao nó “Telegram Sender”. Isto é o output do microfone já descriptado irá ser enviado para o “chatID” atribuído ao nosso Bot” com o tipo “message” para este ser enviado em formato mensagem para o telegram.

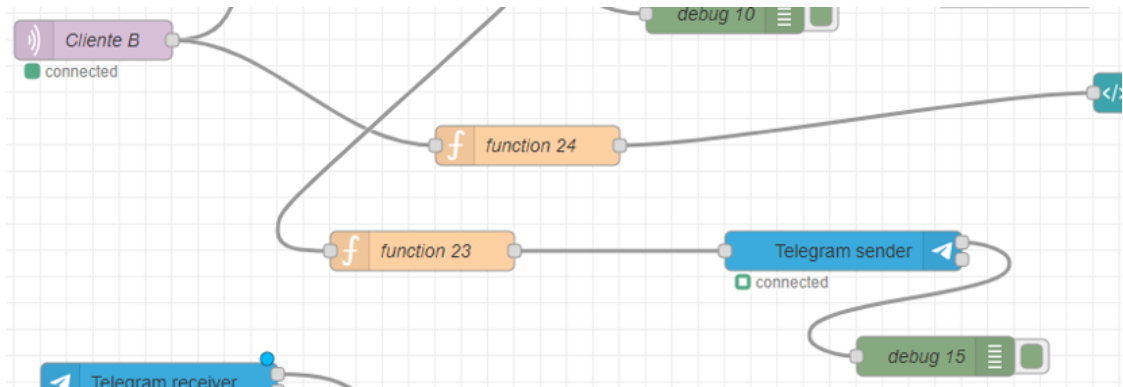


Imagem 24: Função 23 que está conectada ao cliente B

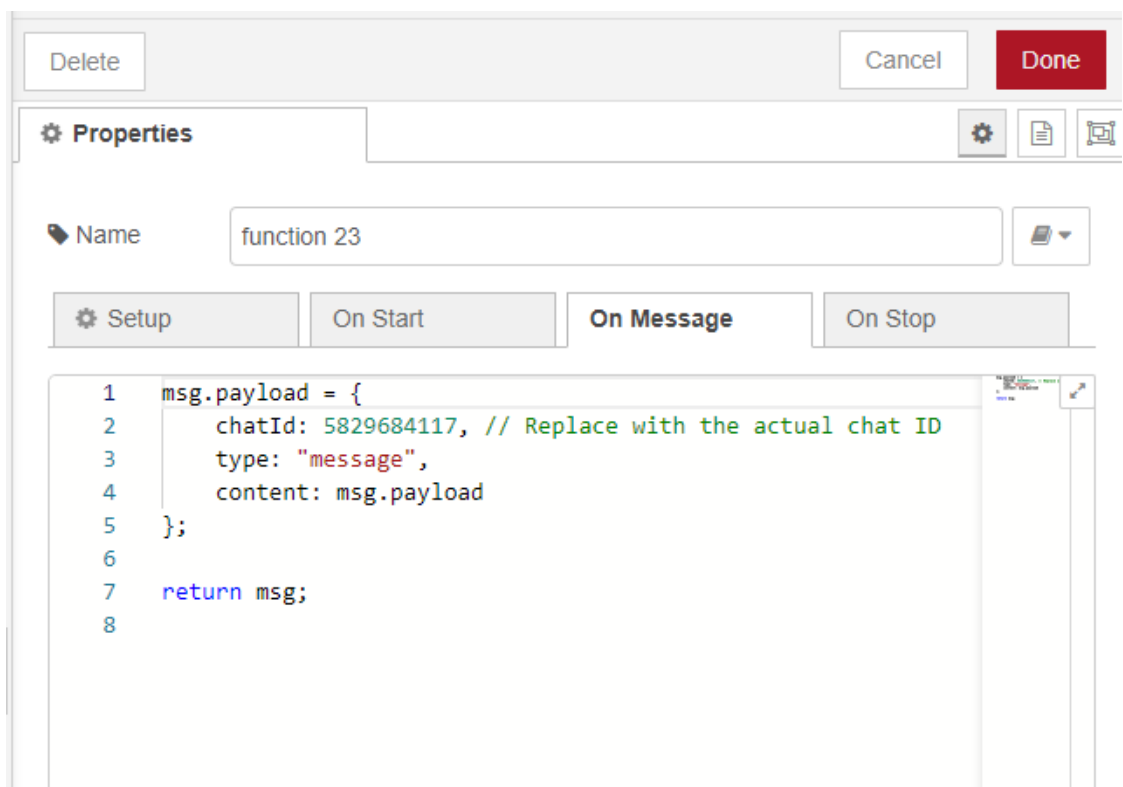


Imagem 25: Função 23 que está conectada ao cliente B

Após isto a mensagem é então enviada para o nosso Bot no telegram:

```
01/06/2023, 01:21:24 node: debug 15
msg.payload : Object
  { chatId: 5829684117, type: "message",
    content: object, options: object,
    sentMessageId: 62 }
```

Imagem 26: Mensagem enviada para o telegram.



Imagem 27: Output apresentado no telegram

5. Depois temos o nó “telegram receiver” que recebe o conteúdo do telegram para ser apresentado no cliente A no Node-RED. Este está também conectado ao node “function 30”

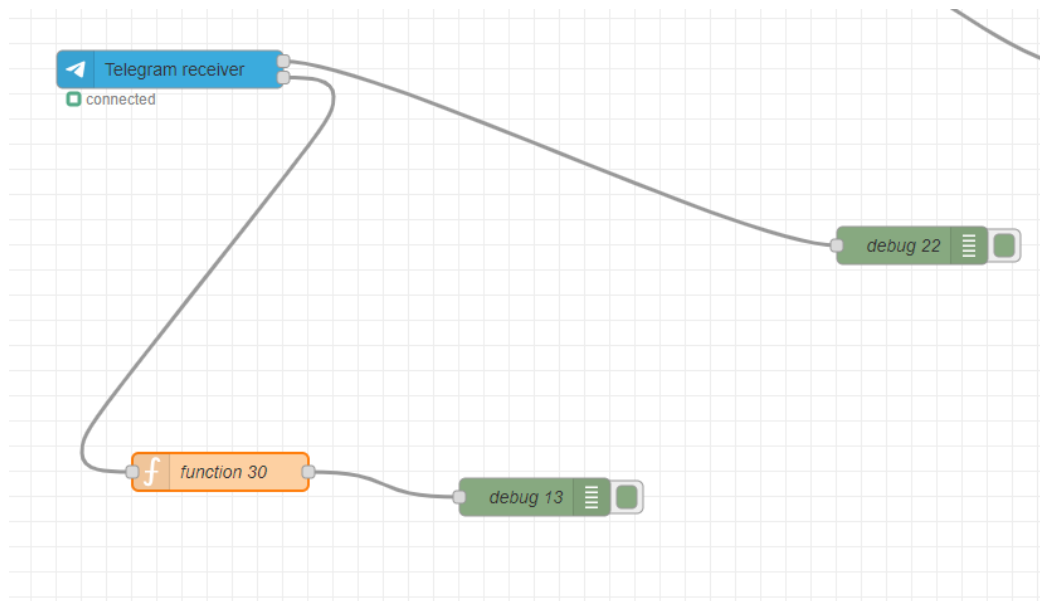


Imagem 28: Nó “Telegram receiver”

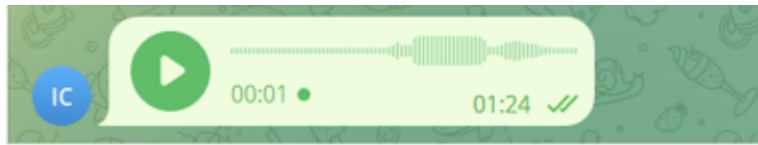


Imagem 29: Envio da mensagem áudio do telegram para o node-red

```
01/06/2023, 01:24:46 node: debug 13
msg.payload : Object
  ▶ { chatId: 5829684117, messageId: 63,
    type: "voice", content:
      "AwACAgQAAxkBAAM_ZHflThNpJNjN30...",
    caption: undefined ... }
```

Imagem 30: Output da mensagem áudio do telegram para o node-red

É possível observar, com o output do debug 13 , o chatId:5829684117 que é o id do chat do nosso Bot,

tipo “voice” e o content.

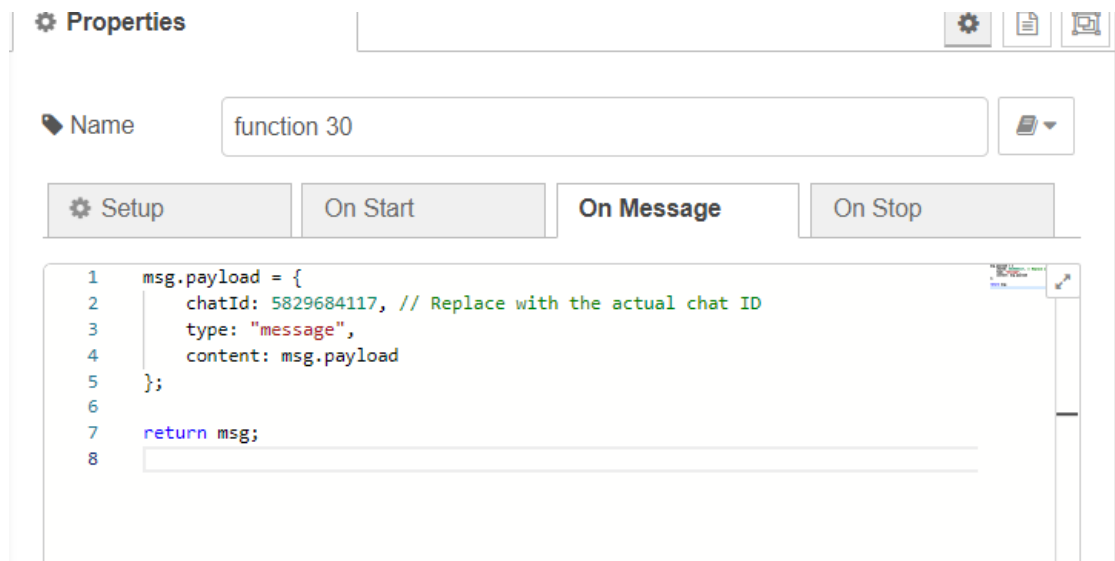


Imagem 31: Função 30 que permite identificar o chatID com o tipo “message” usando o content do msg.payload

8. Fluxo do Node-RED Atualizado após apresentação. (Introdução de áudios armazenados e InfluxDB)

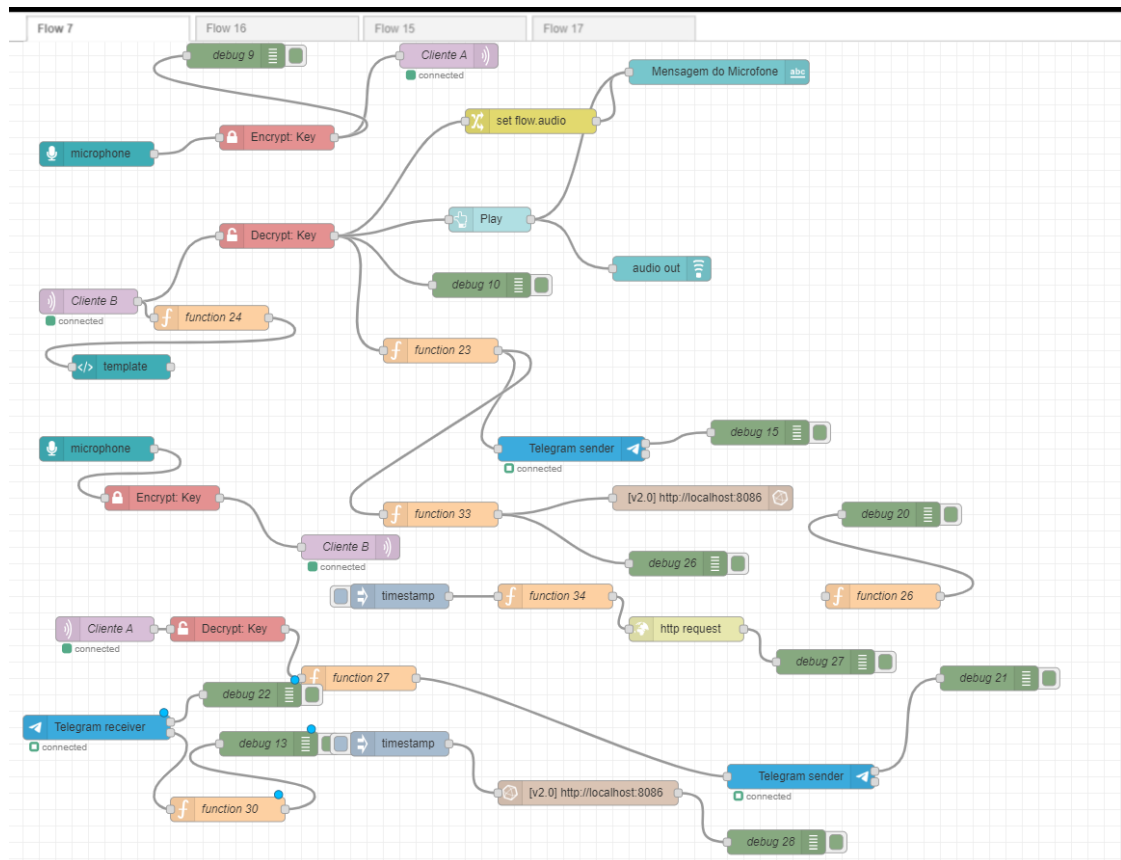


Imagem 32: Fluxo Atualizado

Fizemos a introdução dos nodes do “InfluxDB” que nos permite armazenar os áudios e eliminar o 11º.

Configuramos o nosso InfluxDB da seguinte forma:

Nome do Bucket: Audio

Organização: AudioChat

Abaixo encontra-se uma imagem do nosso bucket e respetivos filtros:

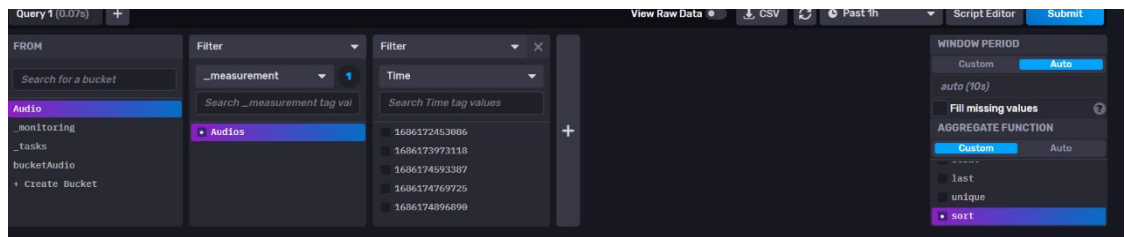


Imagem 33: Bucket “Audio” com os filtros “Audios”

	_start	_stop	_time	_value	Time	_field	_measurement
result = sort	Time = 1686173973118					AUDIO	_measurement
result = sort	Time = 1686174593387					AUDIO	_measurement
result = sort	Time = 1686174769725					AUDIO	_measurement
result = sort	Time = 1686174896890					AUDIO	_measurement

Imagem 34: Áudios Armazenados. Neste caso o output do microfone : “moon”

```
07/06/2023, 22:55:02 node: debug 21
msg.payload : Object
  object
    chatId: 5829684117
    type: "message"
    content: object
      message_id: 135
      from: object
      chat: object
        id: 5829684117
        first_name: "Internet"
        last_name: "Coisas"
        type: "private"
        date: 1686174895
        text: "Moon"
      options: object
        sentMessageId: 135
```

Imagem 35: Áudio com o output “moon” que foi armazenado no influx como já mostramos.

```
Query 1 (0.07s) +
1 from(bucket: "Audio")
2 |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
3 |> filter(fn: (r) => r["_measurement"] == "Audios")
4 |> sort()
5 |> yield(name: "sort")
```

Imagem 36: Script Editor

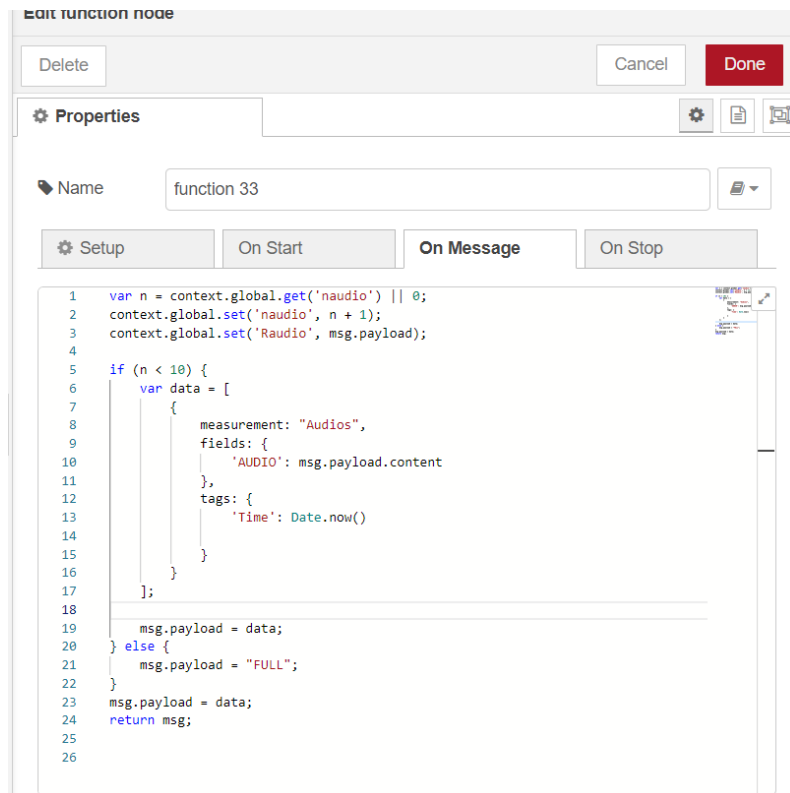


Imagem 37: Função 33 que manda o conteúdo dos áudios para o nosso influxDB

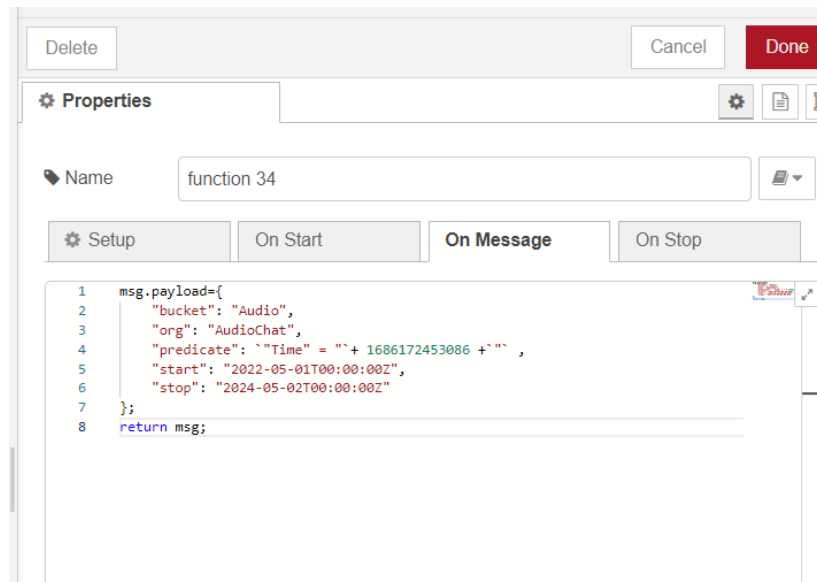


Imagem 38: Função 34

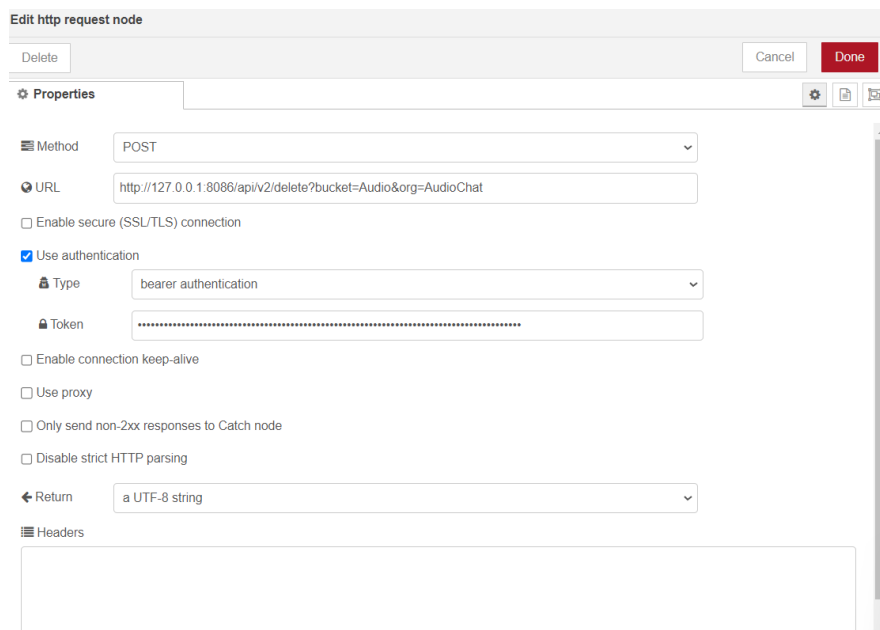


Imagem 39: HTTP Request

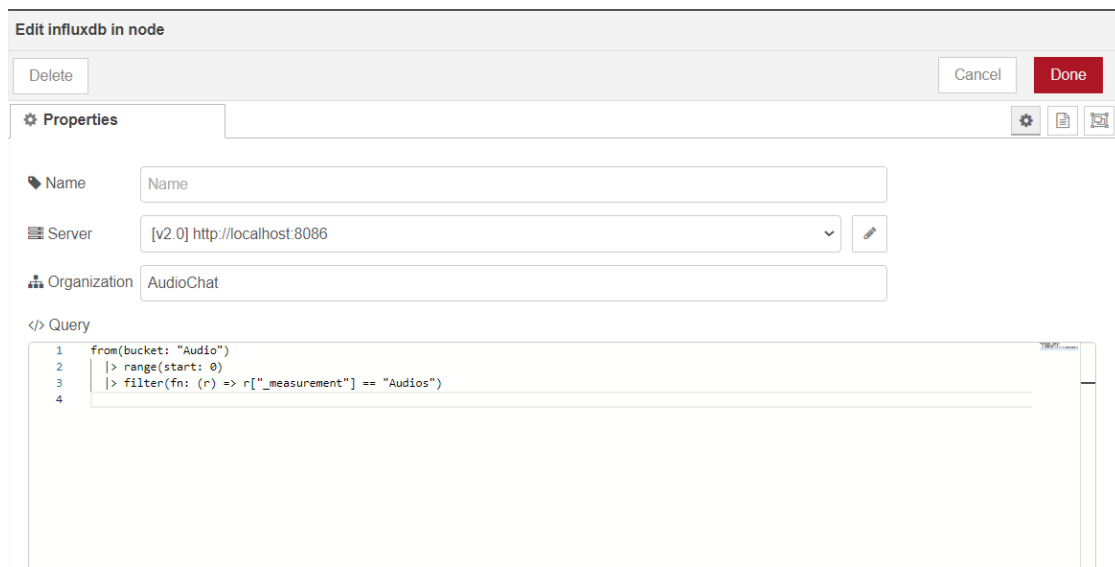


Imagem 40: Acesso aos áudios armazenados

Referencies

1. <https://www.youtube.com/watch?v=wR1zWroRtCg>
2. <https://flows.nodered.org/node/node-red-contrib-telegrambot>
3. <https://www.youtube.com/watch?v=bGzOImDQlo0>
4. <https://homeassistantbrasil.com.br/t/node-red-enviando-mensagens-com-o-telegram/107>
5. <https://www.thesmarthomebook.com/2020/10/13/a-guide-to-using-telegram-with-node-red-and-home-assistant/>
6. <https://wearecommunity.io/communities/india-java-user-group/articles/891>
7. <https://mqtt.org/>
8. <https://aws.amazon.com/pt/what-is/mqtt/>
9. <https://www.hivemq.com/mqtt-essentials/>
10. <https://flows.nodered.org/node/node-red-node-ui-microphone/in/590bc13ff3a5f005c7d2189bbb563976>
11. <https://www.youtube.com/watch?v=BjQpQ9R-CKI>